Day 7: Machine Learning

Kenneth Benoit and Slava Mikhaylov

Introduction to Data Science and Big Data Analytics

25 August 2015

Day 7 Outline

Association rules

Model evaluation in machine learning

- Fitting v. overfitting
- Precision, recall, and accuracy

Naive Bayes

k-Nearest Neighbour

Association rules

Introduction to association rules

- Association rule mining is used to discover objects or attributes that frequently occur together, e.g.
  - movies or music that users prefer
  - baskets of products purchased online or in-store
- Used extensively in recommendation engines
- Terminology:

  transaction a bundle of associated items, such as a collection of movies watched or items puchased, forming the unit of analysis

  itemset the items that make up a transaction, such as purchases, web sites visited, movies watched, etc.

Introduction to association rules

- Association rule mining is used to discover objects or attributes that frequently occur together, e.g.
  - movies or music that users prefer
  - baskets of products purchased online or in-store
- Used extensively in recommendation engines

Introduction to association rules

- Association rule mining is used to discover objects or attributes that frequently occur together, e.g.
  - movies or music that users prefer
  - baskets of products purchased online or in-store
- Used extensively in recommendation engines
- Terminology:

  transaction
  : a bundle of associated items, such as a collection of movies watched or items puchased, forming the unit of analysis

  itemset
  : the items that make up a transaction, such as purchases, web sites visited, movies watched, etc.

Introduction to association rules

- Association rule mining is used to discover objects or attributes that frequently occur together, e.g.
  - movies or music that users prefer
  - baskets of products purchased online or in-store
- Used extensively in recommendation engines
- Terminology:

  transaction a bundle of associated items, such as a collection of movies watched or items puchased, forming the unit of analysis

  itemset the items that make up a transaction, such as purchases, web sites visited, movies watched, etc.
- Many different algorithms, but we will focus on one: the *a priori* algorithm

The *apriori* algorithm

- Two core notions:

  support
  : the support of an item X is the number of transactions that contain X divided by the total number of transactions

  confidence
  : expresses our "confidence" in the relation *if* X*, then* Y
    - formally:

$$\text{support}(\text{union}(X, Y))/\text{support}(X)$$

- The goal is to discover the interesting rules in a dataset above some pre-defined thresholds of support and confidence, such as 10% and 60%)

apriori Rules Example

```
## book data can be found from https://github.com/WinVector/zmPDSwR/tree/master/
# load in book purchase transactions
require(arules, quietly = TRUE, warn.conflicts = FALSE)
(bookbaskets <- read.transactions("~/Dropbox/Classes/LSE Data Science/Lectures/
                                  format = "single", sep = "\t",
                                  cols = c("userid", "title"), rm.duplicates = T

## distribution of transactions with duplicates:
## items
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
## 701 222 106  68  43  39  23  24  18  18  16  10   7   7  13   7   8   5
##  19  20  21  22  23  25  26  27  28  29  30  31  33  34  35  38  39  42
##   3   9   4   4   3   2   2   5   4   5   4   4   1   2   1   1   1   2
##  44  45  47  48  49  52  56  57  59  61  63  71  73  80  84  86  91  93
##   1   1   1   1   1   1   2   1   2   1   2   1   1   1   1   1   1   1
##  95  96  99 103 158 206 260 891
##   1   1   1   1   1   2   1   1
## transactions in sparse format with
##  92108 transactions (rows) and
##  220447 items (columns)
```

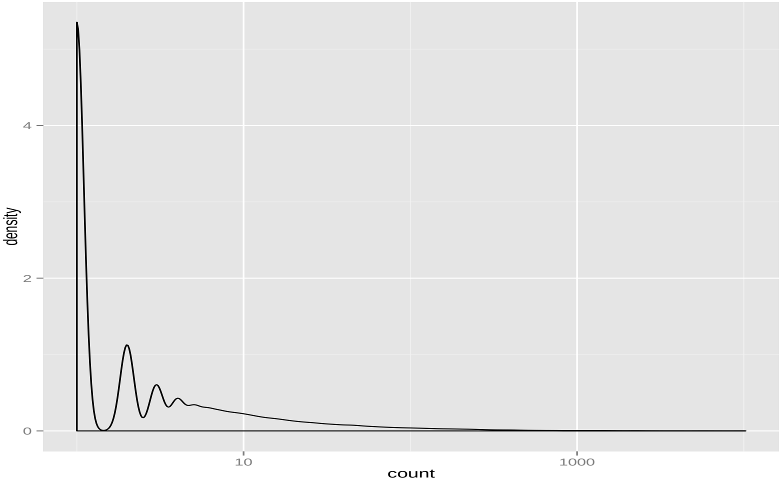
apriori Rules Example

```
# summarize basket sizes
basketSizes <- size(bookbaskets)
summary(basketSizes)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0     1.0     1.0    11.1     4.0 10250.0
```

apriori Rules Example

```
# plot the distribution of basket sizes (log10 scale)
require(ggplot2, quietly = TRUE)
ggplot(data.frame(count = basketSizes)) + scale_x_log10() +
    geom_density(aes(x = count), binwidth = 1)
```



apriori Rules Example

```
# which books are people reading?
bookFreq <- itemFrequency(bookbaskets)
bookCount <- (bookFreq / sum(bookFreq)) * sum(basketSizes)
orderedBooks <- sort(bookCount, decreasing = TRUE)
head(orderedBooks, 10)

##                                      Wild Animus
##                                             2502
##                        The Lovely Bones: A Novel
##                                             1295
##                                She's Come Undone
##                                              934
##                                The Da Vinci Code
##                                              905
##            Harry Potter and the Sorcerer's Stone
##                                              832
##                       The Nanny Diaries: A Novel
##                                              821
##                                  A Painted House
##                                              819
##                             Bridget Jones's Diary
##                                              772
##                          The Secret Life of Bees
##                                              762
## Divine Secrets of the Ya-Ya Sisterhood: A Novel
```

apriori Rules Example

```
# mine the rules using the apriori algorithm
rules <- apriori(bookbaskets_use,
                 parameter = list(support = 0.002, confidence = 0.75))
```

```
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport support minlen maxlen
##        0.75    0.1    1 none FALSE            TRUE   0.002      1     10
##  target   ext
##   rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## apriori - find association rules with the apriori algorithm
## version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[216031 item(s), 40822 transaction(s)] done [0.50s].
## sorting and recoding items ... [1256 item(s)] done [0.03s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 4 5 done [0.03s].
## writing ... [191 rule(s)] done [0.00s].
## creating S4 object  ... done [0.05s].
```

apriori Rules Example

```
summary(rules)

## set of 191 rules
##
## rule length distribution (lhs + rhs):sizes
##   2   3   4   5
##  11 100  66  14
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   3.000   3.000   3.435   4.000   5.000
##
## summary of quality measures:
##     support           confidence          lift
##  Min.   :0.002009   Min.   :0.7500   Min.   : 40.89
##  1st Qu.:0.002131   1st Qu.:0.8113   1st Qu.: 86.44
##  Median :0.002278   Median :0.8468   Median :131.36
##  Mean   :0.002593   Mean   :0.8569   Mean   :129.68
##  3rd Qu.:0.002695   3rd Qu.:0.9065   3rd Qu.:158.77
##  Max.   :0.005830   Max.   :0.9882   Max.   :321.89
##
## mining info:
##             data ntransactions support confidence
##  bookbaskets_use         40822   0.002       0.75
```

apriori Rules Example

```
inspect(head((sort(rules, by = "confidence")), n = 5))

##   lhs                                                rhs
## 1 {Four to Score,
##    High Five,
##    Seven Up,
##    Two for the Dough}                             => {Three To Get Deadly : A
## 2 {Harry Potter and the Order of the Phoenix,
##    Harry Potter and the Prisoner of Azkaban,
##    Harry Potter and the Sorcerer's Stone}         => {Harry Potter and the Ch
## 3 {Four to Score,
##    High Five,
##    One for the Money,
##    Two for the Dough}                             => {Three To Get Deadly : A
## 4 {Four to Score,
##    Seven Up,
##    Three To Get Deadly : A Stephanie Plum Novel,
##    Two for the Dough}                             => {High Five}
## 5 {High Five,
##    Seven Up,
##    Three To Get Deadly : A Stephanie Plum Novel,
##    Two for the Dough}                             => {Four to Score}
```

Model evaluation in machine learning

Generalization and overfitting

- Generalization: A classifier or a regression algorithm learns to correctly predict output from given inputs not only in previously seen samples but also in previously unseen samples
- Overfitting: A classifier or a regression algorithm learns to correctly predict output from given inputs in previously seen samples but fails to do so in previously unseen samples. This causes poor prediction/generalization

How model fit is evaluated

- For discretely-valued outcomes (class prediction): Goal is to maximize the frontier of precise identification of true condition with accurate recall, defined in terms of false positives and false negatives
  - will define formally later
- For continuously-valued outcomes: minimize Root Mean Squared Error (RMSE)

Precision and recall

- Illustration framework

		True condition	
		Positive	**Negative**
Prediction	Positive	True Positive	False Positive (Type I error)
	Negative	False Negative (Type II error)	True Negative

Precision and recall and related statistics

- Precision: $\frac{\text{true positives}}{\text{true positives} + \text{false positives}}$
- Recall: $\frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$

Precision and recall and related statistics

- Precision: $\frac{\text{true positives}}{\text{true positives} + \text{false positives}}$
- Recall: $\frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$
- Accuracy: $\frac{\text{Correctly classified}}{\text{Total number of cases}}$
- $F1 = 2\,\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$
  (the harmonic mean of precision and recall)

Example: Computing precision/recall

Assume:

- We have a sample in which 80 outcomes are really positive (as opposed to negative, as in sentiment)
- Our method declares that 60 are positive
- Of the 60 declared positive, 45 are actually positive

Example: Computing precision/recall

Assume:

- We have a sample in which 80 outcomes are really positive (as opposed to negative, as in sentiment)
- Our method declares that 60 are positive
- Of the 60 declared positive, 45 are actually positive

Solution:

$$\begin{aligned} \text{Precision} &= (45/(45+15)) = 45/60 = 0.75 \\ \text{Recall} &= (45/(45+35)) = 45/80 = 0.56 \end{aligned}$$

Accuracy?

		True condition		
		Positive	Negative	
Prediction	Positive	45		60
	Negative			
		80		

add in the cells we can compute

		True condition		
		Positive	Negative	
Prediction	Positive	45	*15*	60
	Negative	*35*		
		80		

How do we get "true" condition?

- In some domains: through more expensive or extensive tests
- In social sciences: typically by expert annotation or coding
- A scheme should be tested and reported for its reliability

Naive Bayes

Naive Bayes classification

- The following examples refer to “words” and “documents” but can be thought of as generic “features” and “cases”
- We will being with a discrete case, and then cover continuous feature values
- Objective is typically MAP: identification of the *maximum a posteriori* class probability

Multinomial Bayes model of Class given a Word

Consider J word types distributed across I documents, each assigned one of K classes.

At the word level, Bayes Theorem tells us that:

$$P(c_k|w_j) = \frac{P(w_j|c_k)P(c_k)}{P(w_j)}$$

For two classes, this can be expressed as

$$= \frac{P(w_j|c_k)P(c_k)}{P(w_j|c_k)P(c_k) + P(w_j|c_{\neg k})P(c_{\neg k})} \tag{1}$$

Multinomial Bayes model of Class given a Word

Class-conditional word likelihoods

$$P(c_k|w_j) = \frac{P(w_j|c_k)P(c_k)}{P(w_j|c_k)P(c_k) + P(w_j|c_{\neg k})P(c_{\neg k})}$$

- The word likelihood within class
- The maximum likelihood estimate is simply the proportion of times that word j occurs in class k, but it is more common to use Laplace smoothing by adding 1 to each oberved count within class

Multinomial Bayes model of Class given a Word

Word probabilities

$$P(c_k|w_j) = \frac{P(w_j|c_k)P(c_k)}{P(w_j)}$$

- This represents the word probability from the training corpus
- Usually uninteresting, since it is constant for the training data, but needed to compute posteriors on a probability scale

Multinomial Bayes model of Class given a Word

Class prior probabilities

$$P(c_k|w_j) = \frac{P(w_j|c_k)P(c_k)}{P(w_j|c_k)P(c_k) + P(w_j|c_{\neg k})P(c_{\neg k})}$$

- This represents the class prior probability
- Machine learning typically takes this as the document frequency in the training set
- This approach is flawed for scaling, however, since we are scaling the latent class-ness of an unknown document, not predicting class – uniform priors are more appropriate

Multinomial Bayes model of Class given a Word

Class posterior probabilities

$$P(c_k|w_j) = \frac{P(w_j|c_k)P(c_k)}{P(w_j|c_k)P(c_k) + P(w_j|c_{\neg k})P(c_{\neg k})}$$

- This represents the posterior probability of membership in class k for word j

Moving to the document level

- The "Naive" Bayes model of a joint document-level class posterior assumes conditional independence, to multiply the word likelihoods from a "test" document, to produce:

$$P(c|d) = P(c) \prod_j \frac{P(w_j|c)}{P(w_j)}$$

Moving to the document level

- The "Naive" Bayes model of a joint document-level class posterior assumes conditional independence, to multiply the word likelihoods from a "test" document, to produce:

$$P(c|d) = P(c) \prod_j \frac{P(w_j|c)}{P(w_j)}$$

- This is why we call it "naive": because it (wrongly) assumes:
  - *conditional independence* of word counts
  - *positional independence* of word counts

Naive Bayes Classification Example

(From Manning, Raghavan and Schütze, *Introduction to Information Retrieval*)

► **Table 13.1** Data for parameter estimation examples.

	docID	words in document	in $c = China$?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

Naive Bayes Classification Example

Example 13.1: For the example in Table 13.1, the multinomial parameters we need to classify the test document are the priors $\hat{P}(c) = 3/4$ and $\hat{P}(\overline{c}) = 1/4$ and the following conditional probabilities:

$$\begin{aligned}
\hat{P}(\mathsf{Chinese}|c) &= (5+1)/(8+6) = 6/14 = 3/7 \\
\hat{P}(\mathsf{Tokyo}|c) = \hat{P}(\mathsf{Japan}|c) &= (0+1)/(8+6) = 1/14 \\
\hat{P}(\mathsf{Chinese}|\overline{c}) &= (1+1)/(3+6) = 2/9 \\
\hat{P}(\mathsf{Tokyo}|\overline{c}) = \hat{P}(\mathsf{Japan}|\overline{c}) &= (1+1)/(3+6) = 2/9
\end{aligned}$$

The denominators are $(8+6)$ and $(3+6)$ because the lengths of $text_c$ and $text_{\overline{c}}$ are 8 and 3, respectively, and because the constant B in Equation (13.7) is 6 as the vocabulary consists of six terms.

We then get:

$$\begin{aligned}
\hat{P}(c|d_5) &\propto 3/4 \cdot (3/7)^3 \cdot 1/14 \cdot 1/14 \approx 0.0003. \\
\hat{P}(\overline{c}|d_5) &\propto 1/4 \cdot (2/9)^3 \cdot 2/9 \cdot 2/9 \approx 0.0001.
\end{aligned}$$

Thus, the classifier assigns the test document to c = *China*. The reason for this classification decision is that the three occurrences of the positive indicator Chinese in d_5 outweigh the occurrences of the two negative indicators Japan and Tokyo.

Naive Bayes with continuous covariates

```r
library(e1071)   # has a normal distribution Naive Bayes

# Congressional Voting Records of 1984 (abstentions treated as missing)
data(HouseVotes84, package = "mlbench")
model <- naiveBayes(Class ~ ., data = HouseVotes84)

# predict the first 10 Congresspeople
data.frame(Predicted = predict(model, HouseVotes84[1:10,-1]),
           Actual = HouseVotes84[1:10,1],
           postPr = predict(model, HouseVotes84[1:10, -1], type = "raw"))
```

```
##     Predicted     Actual postPr.democrat postPr.republican
## 1  republican republican    1.029209e-07      9.999999e-01
## 2  republican republican    5.820415e-08      9.999999e-01
## 3  republican   democrat    5.684937e-03      9.943151e-01
## 4    democrat   democrat    9.985798e-01      1.420152e-03
## 5    democrat   democrat    9.666720e-01      3.332802e-02
## 6    democrat   democrat    8.121430e-01      1.878570e-01
## 7  republican   democrat    1.751512e-04      9.998248e-01
## 8  republican republican    8.300100e-06      9.999917e-01
## 9  republican republican    8.277705e-08      9.999999e-01
## 10   democrat   democrat    1.000000e+00      5.029425e-11
```

Overall prediction performance

```
# now all of them: this is the resubstitution error
(mytable <- table(predict(model, HouseVotes84[,-1]), HouseVotes84$Class))

##
##              democrat republican
##   democrat        238         13
##   republican       29        155

prop.table(mytable, margin=1)

##
##                democrat republican
##   democrat   0.94820717 0.05179283
##   republican 0.15760870 0.84239130
```

With Laplace smoothing

```
model <- naiveBayes(Class ~ ., data = HouseVotes84, laplace = 3)
(mytable <- table(predict(model, HouseVotes84[,-1]), HouseVotes84$Class))

##
##              democrat republican
##   democrat        237         12
##   republican       30        156

prop.table(mytable, margin=1)

##
##                democrat republican
##   democrat   0.95180723 0.04819277
##   republican 0.16129032 0.83870968
```

k-Nearest Neighbour

k-nearest neighbour

- A non-parametric method for classifying objects based on the training examples taht are *closest* in the feature space

k-nearest neighbour

- A non-parametric method for classifying objects based on the training examples taht are *closest* in the feature space
- A type of instance-based learning, or "lazy learning" where the function is only approximated locally and all computation is deferred until classification

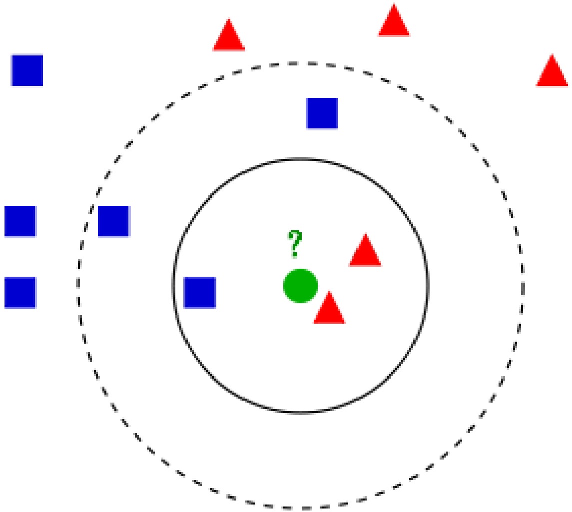
k-nearest neighbour

- A non-parametric method for classifying objects based on the training examples taht are *closest* in the feature space
- A type of instance-based learning, or “lazy learning” where the function is only approximated locally and all computation is deferred until classification
- An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common amongst its k nearest neighbors (where k is a positive integer, usually small)

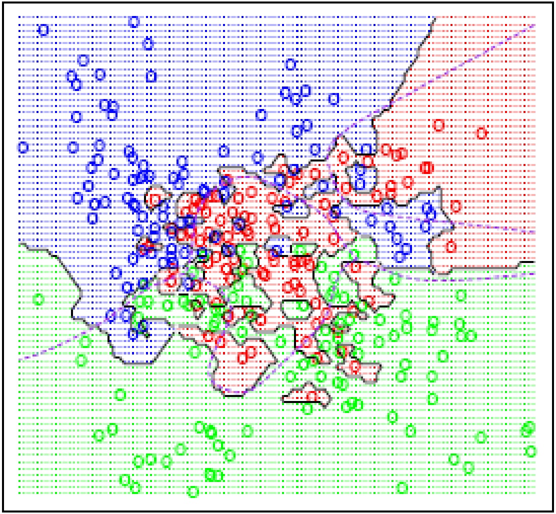
k-nearest neighbour

- A non-parametric method for classifying objects based on the training examples taht are *closest* in the feature space
- A type of instance-based learning, or "lazy learning" where the function is only approximated locally and all computation is deferred until classification
- An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common amongst its k nearest neighbors (where k is a positive integer, usually small)
- Extremely *simple*: the only parameter that adjusts is k (number of neighbors to be used) - increasing k *smooths* the decision boundary

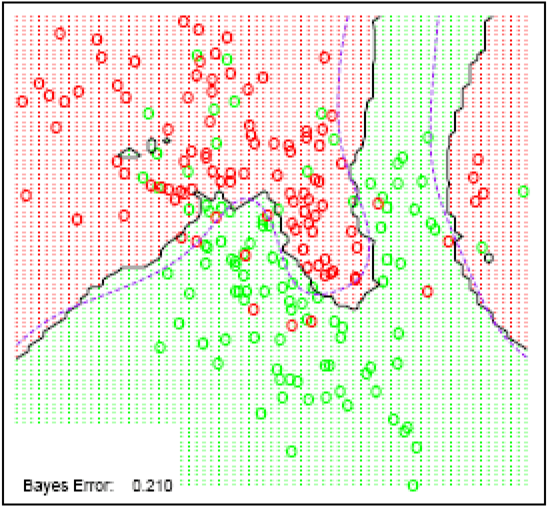
k-NN Example: Red or Blue?



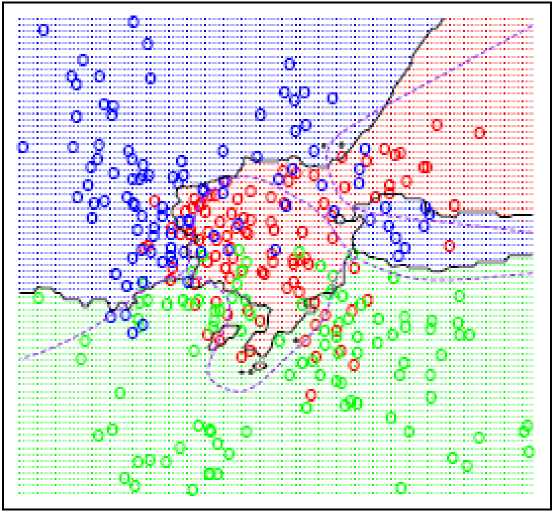
$k = 1$



$k = 7$



$k = 15$



Classifying amicus curiae briefs (Evans et al 2007)

```r
## kNN classification
require(class)

## Loading required package:  class

require(quanteda, warn.conflicts = FALSE, quietly = TRUE)
require(quantedaData, quietly = TRUE)
data(amicusCorpus)
# create a matrix of documents and features
amicusDfm <- dfm(amicusCorpus, ignoredFeatures=stopwords("english"),
                 stem=TRUE, verbose=FALSE)
# threshold-based feature selection
amicusDfm <- trim(amicusDfm, minCount=10, minDoc=20)

## Features occurring less than 10 times: 9122
## Features occurring in fewer than 20 documents: 10572
```

Classifying amicus curiae briefs (Evans et al 2007)

```
# tf-idf weighting
amicusDfm <- weight(amicusDfm, "tfidf")

## Note:  method with signature 'CsparseMatrix#Matrix#missing#replValue'
chosen for function '[<-',
##  target signature 'dfmSparse#ngCMatrix#missing#numeric'.
##  "Matrix#nsparseMatrix#missing#replValue" would also be valid

# partition the training and test sets
train <- amicusDfm[!is.na(docvars(amicusCorpus, "trainclass")), ]
test  <- amicusDfm[!is.na(docvars(amicusCorpus, "testclass")), ]
trainclass <- docvars(amicusCorpus, "trainclass")[1:4]
```

Classifying amicus curiae briefs (Evans et al 2007)

```r
# classifier with k=1
classified <- knn(train, test, trainclass, k=1)
table(classified, docvars(amicusCorpus, "testclass")[-c(1:4)])

##
## classified AP AR
##          P 14  5
##          R  5 74
```

Classifying amicus curiae briefs (Evans et al 2007)

```
# classifier with k=2
classified <- knn(train, test, trainclass, k=2)
table(classified, docvars(amicusCorpus, "testclass")[-c(1:4)])

##
## classified AP AR
##          P 13 47
##          R  6 32
```

k-nearest neighbour issues: Dimensionality

- Distance usually relates to all the attributes and assumes all of them have the same effects on distance
- Misclassification may results from attributes not confirming to this assumption (sometimes called the "curse of dimensionality") – solution is to reduce the dimensions
- There are (many!) different *metrics* of distance